# VIRTUAL MOUSE

## A PROJECT REPORT

***Submitted by***

|  |  |
| --- | --- |
| **DEEPIKA P** | **211418104046** |
| **JANANI N** | **211418104094** |
| **JANET JESSICA R** | **211418104095** |

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

## COMPUTER SCIENCE AND ENGINEERING



**PANIMALAR ENGINEERING COLLEGE**

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

**MAY 2022**

# PANIMALAR ENGINEERING COLLEGE

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

**BONAFIDE CERTIFICATE**

Certified that this project report **“VIRTUAL MOUSE”** is the bonafide work of **DEEPIKA P [REGISTER NO: 211418104095], JANANI N [REGISTER NO:211418104095], JANET JESSICA R [REGISTER NO: 211418104095]** who

carried out the project under my supervision.

## SIGNATURE SIGNATURE

**Dr.S.MURUGAVALLI,M.E.,Ph.D., Dr. L. JABASHEELA,M.E.,Ph.D., HEAD OF THE DEPARTMENT SUPERVISOR**

## PROFESSOR

DEPARTMENT OF CSE, DEPARTMENT OF CSE,

PANIMALAR ENGINEERING COLLEGE, PANIMALAR ENGINEERING COLLEGE, NASARATHPETTAI, NASARATHPETTAI,

POONAMALLEE, POONAMALLEE,

CHENNAI-600 123. CHENNAI-600 123.

Certified that the above candidates were examined in the Anna University Project Viva- Voce Examination held on...........................

## INTERNAL EXAMINER EXTERNAL EXAMINER

**DECLARATION BY THE STUDENT**

We DEEPIKA P [REGISTER NO:211418104046], JANANI N [REGISTER NO:211418104094], JANET JESSICA R [REGISTER NO:

211418104095] hereby declare that this project report titled “ VIRTUAL MOUSE” under the guidance of Dr. L. JABASHEELA is the original work done by us and we have not plagiarized or submitted to any other degree in any other university by us.

1.

2.

3.

**ACKNOWLEDGEMENT**

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI**, **Dr.C.SAKTHI KUMAR,M.E.,Ph.D** and **Dr.SARANYA SREE SAKTHI**

**KUMAR B.E.,M.B.A.,Ph.D.,** for providing us with the necessary facilities to undertake this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr. S. MURUGAVALLI, M.E.,Ph.D.,**

for the support extended throughout the project.

We would like to thank our Project Guide **Dr. L. JABASHEELA, M.E., Ph,D.,** and all the faculty members of the Department of CSE for their advice and encouragement and parents for the successful completion of the project.

## DEEPIKA P JANANI N

**JANET JESSICA R**

## ABSTRACT

Virtual Mouse utilizes various image processing techniques to eliminate the need for a hardware mouse while enabling users to interact with the computer system through webcam. It enhances Human Computer Interaction (HCI) by utilizing a camera and computer vision technology to control numerous mouse events and is capable of performing all functions that a physical computer mouse can accomplish. This type of HCI is referred to as "natural user interface" (NUI). In recent decades, the keyboard and mouse have become progressively important in human-computer interaction. This incorporates the advancement of touch technology over buttons, as well as other gesture control modalities. A conventional camera may be utilized to create a hand tracking- based virtual mouse. We combine camera and computer vision technologies, such as finger-node identification and gesture recognition, into the proposed system to handle mouse operations (volume control, right click, left click, drag folders), and demonstrate how it can do all of the functions of existing mouse devices. In this, hand gesture tracking is generated through the detection of the contour and formation of a convex hull around it. The area ratio of the contour and the hull created are used to extract hand features. This technique is put to numerous tests in real-world scenarios.

|  |  |  |
| --- | --- | --- |
| **TABLE NO** | **LIST OF TABLES**  **TABLE DESCRIPTION** | **PG NO.** |
| 4.2 | Data Dictionary | 12 |
| 7.1 | Accuracy Calculation of virtual mouse | 45 |

|  |  |  |
| --- | --- | --- |
| **FIG NO** | **LIST OF FIGURES**  **DESCRIPTION** | **PG NO** |
| 4.1 | ER Diagram | 11 |
| 4.3 | System flow diagram | 14 |
| 4.4 | Use case diagram | 15 |
| 4.5 | Class diagram | 16 |
| 4.6 | Activity Diagram | 17 |
| 4.7 | Sequence Diagram | 18 |
| 5.1 | System Architecture | 19 |
| A1.1 | Copy & Paste in voice assistant | 48 |
| A1.2 | Date & Time in voice assistant | 48 |
| A1.3 | Exit in voice assistant | 49 |
| A1.4 | File Navigation in voice assistant | 49 |
| A1.5 | Google search in voice assistant | 50 |
| A1.6 | Launch/Stop Gesture recognition | 50 |
| A1.7 | Neutral Gesture | 51 |
| A1.8 | Move cursor | 51 |
| A1.9 | Right Click | 52 |
| A1.10 | Brightness Control | 52 |
| A1.11  A1.12 | Click and drag  Double click | 53  53 |

## LISTOFABBREVIATIONS AND SYMBOLS

**HCI –** Human Computer Interaction

**RGB** – Ged Green Blue

**NUI** – Natural User Interface

**Open CV** – Opensource Computer Vision

**CMD-** Command(s)

**UI**- User Interface

**SAPI**- Speech Application Programming Interface

**COM**- Communication

**GUI**- Graphical User Interface

**AI**- Artificial Intelligence

**NLP**- Natural Image Processing

**TTS**- Text To Speech

**PYTTSX-** Python Text To Speech

* Actor
* Initial State
* Final State



* State

State

## TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT** | iii |
|  | **LIST OF TABLES** | iv |
|  | **LIST OF FIGURES** | v |
|  | **LIST OF ABBREVIATIONS & SYMBOLS** | vi |
| **1.** | **INTRODUCTION**  1.1 Overview | 1 |
|  | 1.2 Problem Definition | 2 |
| **2.** | **LITERATURE SURVEY** | 3 |
| **3.** | **SYSTEM ANALYSIS** |  |
|  | 3.1 Existing System | 7 |
|  | 3.2 Proposed system | 7 |
|  | 3.3 Feasibility Study | 8 |
|  | 3.4 Hardware Environment | 10 |
| 3.5 Software Environment 10 | | |
| **4.** | **SYSTEM DESIGN** |  |
|  | 4.1. ER diagram 11 | |
|  | 4.2 Data dictionary 12 | |
|  | 4.3 System Flow Diagram | 14 |
|  | 4.4 Use case diagram | 15 |

**CHAPTER NO. TITLE PAGE NO.**

* 1. [Class Diagram](#_TOC_250008) 16
  2. [Activity Diagram](#_TOC_250007) 17
  3. [Sequence Diagram](#_TOC_250006) 18

1. SYSTEM ARCHITECTURE
   1. [System Architecture Overview](#_TOC_250005) 19
   2. [Module Design Specification](#_TOC_250004) 20
   3. [Algorithm](#_TOC_250003) 22
2. SYSTEM IMPLEMENTATION
   1. [Coding](#_TOC_250002) 25
3. PERFORMANCE ANALYSIS 44
4. CONCLUSION
   1. Conclusion and Future Enhancements 47

[APPENDICES](#_TOC_250001)

[A.1 Sample Screens](#_TOC_250000) 48

References

55

## CHAPTER – 1 INTRODUCTION

* 1. **OVERVIEW**

Vision-based hand gesture recognition includes static hand gesture recognition and dynamic hand gesture recognition. Dynamic hand gestures are more reliable and natural in human-robot interaction (HRI) or human-computer interaction (HCI) than static hand gestures (HCI). Dynamic hand gestures, on the other hand, combine both temporal and spatial information, making recognition more difficult. Furthermore, in order to recognize dynamic hand gestures online, each dynamic hand motion must be located and segmented from streaming video. Therefore, developing effective dynamic hand gesture localization methods is an important research challenge. Our study on online detection and recognition of dynamic hand movements, design a new recognition method, and apply it to hand-based HRI in order to address these issues. The principal method of hand gesture recognition uses a feature extractor to extract hand gesture features and then a using classifier to classify the extracted features for the recognition of different hand gestures. Image-based hand gesture detection has made significant progress recently. Furthermore, 3D pose estimation of hand joints can increase the accuracy of dynamic hand gesture recognition. Because the hand often occupies only a small part of a collected image, the accurate detection of dynamic hand gestures is the key to dynamic hand gesture recognition. This device, the webcam, will be constantly used by software that monitors the user’s gestures in order to interpret and translate them into pointes motion, comparable to a physical mouse.

## PROBLEM DEFINITION

The physical mouse that we use in our daily lives will ultimately wear down due to the movement of the mechanical components because it is constructed of moving parts. It’s no surprised that every technological devices have its own limitations, especially when it comes to computer devices. In addition to a flat working surface, using a physical mouse requires the installation of specialized gear. All these are required to operate the mouse. The duration a mouse lasts for you may be impacted by whether it is wired or wireless. The degree to which the performance of a physical mouse is dependent on the environment in which it is used makes it difficult to adapt to new configurations. As a result of this dependence, it can be challenging to adapt to new configurations.

Even in its current state of operation, the mouse is only capable of performing a small subset of all of the possible functions that it is capable of. In this project we are trying to eliminate the use of a physical mouse. After the review of various type of the physical mouse, the problems are identified and generalized. Mechanical wear and tear are experienced by physical mouse. To utilize a physical mouse, you’ll need appropriate hardware and a flat surface. Physical mouse are difficult to adapt to varied surroundings, and their performance varies depending on the situation. Even in today’s operating conditions, the mouse has limited capabilities. Every wired and wireless mouse has a different life time

## CHAPTER 2 LITERATURE SURVEY

1. **Banda Aneela , Narlanka Vasuki, Rebelli Sai Sahana, Charupalli Sunil Kumar, (2021), ”Implementing a Real Time Virtual Mouse System and Fingertip Detection based on Artificial Intelligence”. pp. 6299–6308[1]**

It involves the progression of touch technology over buttons and variety of other gesture control modalities. A normal camera can be used to construct a hand tracking- based virtual mouse application. Here they combined camera and computer vision technologies, such as finger- tip identification and gesture recognition, into the proposed system to handle mouse operations (volume control, right click, left click), and show how it can execute all that existing mouse devices can but it does not work on browser.

## [Sherin Mohammed Sali Shajideen](https://ieeexplore.ieee.org/author/37086874147); [VH Preetha](https://ieeexplore.ieee.org/author/37086054925)(2019) “Hand Gestures – Virtual Mouse for Human Computer Interaction”, pp. 5239–2301.[2]

It focuses on the improvement of human computer interaction systems using hand gesture with 3-D space by using two camera in position. It works on accurate values in finger pointing with 3-D space methodology. It detects complete hand with palm and fingers but the process is too complex.

## Vantukala VishnuTeja Reddy, Thumma Dhyanchand , Galla Vamsi Krishna , Satish Maheshwaram (2019), “Virtual Mouse Control Using hand gloves ” (IEEE), pp. 1239–5239.[3]

This paper involves a virtual mouse system using colored hand glove based on HCI using computer vision and hand gestures. Gestures captured with a webcam which

processed with color segmentation, detection technique and feature extraction using robotic arm technology.

## Tiwari, Siddhi Parkar, Shruti Gharat, Kinjal Patel, Prof. Khalil Pinjari, (2019) “Design and Development of Hand Gesture Based Virtual Mouse”, Institute of Electrical and Electronics Engineering(IEEE) , pp. 8745–3219.[4]

It proposes a virtual mouse system based on HCI using computer vision and hand gestures. Gestures captured with a built-in camera or webcam were processed with color segmentation & detection technique. It works on Histogram analysis of skin pixels methodology

## Kollipara Sai Varun, I. Puneeth, Dr. T. Prem Jacob, (2019) “Virtual Mouse Implementation using Open CV”, Institute of Electrical and Electronics Engineering(IEEE), pp. 2341–9834.[5]

This paper proposes the controlling of system by showing and detects using Real-time computer vision, Deep Learning technology. This system can able to identify fingers more accurate but it works only for basic functions.

## Swati Tiwari, Siddhi Parkar, Shruti Gharat, Kinjal Patel, Prof. Khalil Pinjari, (2018), “Virtual mouse using sixth sense technology” Institute of Electrical and Electronics Engineering(IEEE), pp. 4312–7865.[6]

This paper has proposed the idea of virtual mouse using sixth sense technology since it uses gestures for its interaction making it highly responsive in real time. This paper uses GSM module for sending and fetching information. In this paper color markers are used such as RGB for HSV conversion which then later uses for feature extraction and image processing. The disadvantage of this paper is that it uses external color markers which is not feasible and accessible.

## Sherin Mohammed Sali Shajideen Preetha V H, YEAR: March 2018, “Virtual Mouse for Human Computer Interaction”, Institute of Electrical and Electronis Engineering(IEEE), ISBN : pp. 7891–2308.[7]

This paper focuses on the improvement of human computer interaction systems using hand gesture with 3-D space by using two camera in position. The hand pointing gesture is estimated and mapped to the screen coordinate system. The human-computer interaction (HCI) system proposed consists of hand pointing detection. The overview of the system has three different strategies for hand pointing gesture such as detection of hand region, tracking of hand features, Making 3D pointing towards direction.

## Changhyun jeon, oh-jin kwon, dongil shin, and dongkyoo shin, (2018), “Hand-Mouse Interface Using Virtual Monitor Concept for Natural Interaction”, Institute of Electrical and Electronics Engineering(IEEE), pp. 6534–2354.[8]

This paper uses implementation of hand mouse interface that introduces a concept called as “virtual monitor” to extract the user’s physical features through Kinect in real time because recognition algorithms such as NUI/NUX are difficult to train and takes a long time for testing. In this paper, preprocessing, normalization and feature extraction are used. The virtual monitor uses a virtual space to control the hand mouse. The accuracy experiment showed the high accuracy level of the mouse functions [drag (80.9%), click (80%), double-click (76.7%)].

## Itsuo Kumazawa, Toshihiro Kai, Yoshikazu Onuki, and Shunsuke Ono,(2017) “Measurement of 3D-Velocity by High-Frame-Rate Optical Mouse Sensors to Extrapolate 3D Position Captured”, Institute of Electrical and Electronis Engineering(IEEE), pp. 7634-1254.[9]

This paper proposes to use a pair of optical-mouse-sensors as a stereo image sensor to measure 3D velocity and use it to extrapolate 3D position measured by a low frame-rate stereo camera. This paper specifies that it requires a lot of computational costs to find correspondence between stereo images to compute distance. This paper deals with the optical mouse sensor mounted with lenses customized for quick hand motion detection. A quick hand swing is performed within 66ms for the range of 60 degree of angle over the sensor. 30 trials for each of 8 swing directions are conducted under various illumination conditions.

## Upasana, V Monisha Joseph, Kanchana V.p , YEAR : May,2017 “Virtual Mouse with RGB Colored Tapes”, Institute of Electrical and Electronics Engineering(IEEE), pp. 2319–5412.[10]

This paper tries to use a camera and computer vision technologies such as gesture recognition and image segmentation to control tasks with colored tapes and shows how it can perform all the mouse functions a physical mouse can perform. In the first phase, hand gestures are acquired using a camera based on colour detection technique using segmentation and image subtraction algorithm. In the second phase, RGB colored tapes are used to control different functions of the mouse and also the combination of these 3 colors by considering the area of each object/colored tape using the blob analysis and the bounding box algorithm. The user must wear the red, green and blue tapes to the fingers such that it is easy to make the movements for each tape and also the combination of the colored tapes to acquire the desired output of the cursor movement in the system

**CHAPTER 3 SYSTEM ANALYSIS**

## 3.1. EXISTING SYSTEM

The currently available system for controlling the monitor consists of a generic mouse and trackpad, and there is no hand gesture system available. Remote access to the monitor screen is not possible using hand gestures. The scope is restricted to the virtual mouse field, despite the fact that it attempts to implement the huge majority of it. Hand recognition-assisted basic mouse activities make up the present virtual mouse control system. This enabled us to move the mouse cursor across the screen, click left or right, drag, and perform other basic mouse activities. Because hand recognition was not used further, this alternative was not pursued. They used a technique known as static hand recognition, which involves simply recognizing the shape of the hand and defining an action for each shape. Then there was an external hand glove that required additional hardware, so it was dropped. Then there was the recognition of color tapes, where the color was recognized, but the major drawback was that it began to recognize the background colors as well.

## PROPOSED SYSTEM

Using the existing system, even though there are a number of quick access methods available for the hand and mouse gesture for the laptops, using our project, we could make use of the laptop or web-cam and by recognizing the hand gesture, we could control mouse and perform basic operations such as controlling the mouse pointer, selecting and deselecting using left click. Alternatively, we could make use of the web- cam to control the mouse. Python is a simple language that is platform independent with flexibility and portability, all of which are desirable in creating a program that is focused in such an aim as creating a virtual mouse and hand recognition system. The system that we are implementing, which has been written in python code, will be significantly more responsive and will be much easier to implement because python is a simple language.

## FEASIBILITY STUDY

* + - Economical
    - Technical
    - Social

## ECONOMICAL FEASIBILITY

Here we have done our project estimation by COCOMO model based on total Lines of Code (LOC) which required to develop our project. Total Lines of code (LOC)=850

|  |  |
| --- | --- |
| Module | Category |
| Hand gesture recognition | Organic |
| Voice Assistant | Organic |
| Virtual Mouse | Semi-Detached |

Hand Gesture Recognition: (LOC=540) Estimation of development effort = a(KLOC)b PM

= 2.4(0.54)1.05 PM

= 1.36 PM

Estimation of Development time = c(effort)d

= 2.5(1.26)0.38

= 2.72 M

VOICE ASSISTANT:

Lines of Code (LOC)=250 Estimation of Effort = 2.4(0.25)1.05

=0.5 PM

Estimation of Development Time = 2.5(0.5)0.38

= 1.9 M

VIRTUAL MOUSE:

Lines of Code (LOC)= 56 Estimation of Effort= 3.0(0.056)1.2

= 0.09 PM

Estimation of Development Time = 2.3(0.09)0.35

= 1.07 M

Total Effort= 1.26+0.5+0.09=1.85 PM

Total Development Time=2.72+1.9+1.07= 5.6 months

## TECHNICAL FEASIBILITY

Language used- Python

Our system is implemented using python programming language,since Python has inbuilt modules like cv2, Media pipe and Autopy which are all required for tracking and controlling the movements of hand.

System used- Pycharm

We have used Pycharm platform for running our task since it is an integrated development environment used in computer programming specifically for the Python programming language.

## SOCIAL FEASIBILITY

As the name virtual mouse denotes this project is to eliminate the needs of having a physical mouse while able to interact with the computer system through webcam by using various image processing techniques. This virtual mouse project will benefit many ranges of society which includes the physically disabled because our project eliminates the need of a physical mouse. Overall feasibility study of the project reveals that the goals of the proposed system are achievable.

## HARDWARE REQUIREMENTS

* + - Processor -i3
    - RAM- 4 GB
    - Hard-disk- 500 MB
    - Webcam
    - Speakers
    - Microphone

## SOFTWARE REQUIREMENTS

* + - OS- Windows
    - Server- PyCharm
    - Scripting Language- Python

## ER DIAGRAM

**CHAPTER 4 SYSTEM DESIGN**

ER diagram is a graphical approach for designing a database. It is a high-level data model that defines data elements and their relationship for a system. An ER model is used to represent real-world objects in the form of entities and attributes. In the above representation, there are two entities such as voice assistant and the virtual mouse depicted through rectangle box. Both the entities are related through diamond box. The attributes of corresponding entities are depicted through ovals.

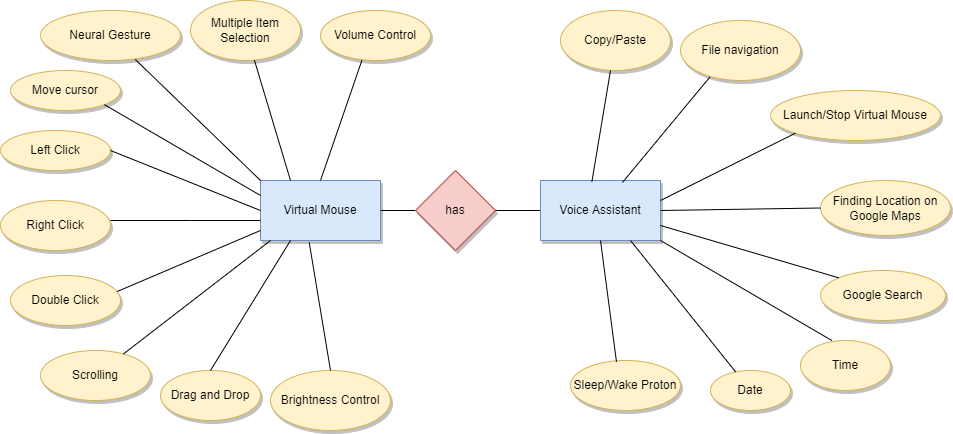


Fig 4.1 ER diagram of Virtual mouse & Voice assistant

## DATA DICTIONARY

The below data dictionary table explains the gestures and voice commands used in the virtual mouse

|  |  |  |
| --- | --- | --- |
| **S. No** | **Gesture** | **Description** |
| 1. | Neural Gesture | Used to halt/stop execution of current gesture. |
| 2. | Move Cursor | Cursor is assigned to the midpoint of index and middle fingertips. Speed of the cursor movement is proportional  to the speed of hand. |
| 3. | Multiple item selection | This gesture moves the cursor to the desired location. Speed of the cursor movement is proportional to the  speed of hand. |
| 4. | Left Click | Index finger is used in the motion of a tap for left click. |
| 5. | Right Click | Middle finger is used in the motion of a tap for right  click. |
| 6. | Double Click | Index and middle finger are joined together for double  click. |
| 7. | Scrolling | Vertical and Horizontal scrolls are controlled by vertical and horizontal pinch movements respectively. |
| 8. | Drag and Drop | Can be used to move/transfer files from one directory to  other. |
| 9. | Brightness Control | The rate of increase/decrease of brightness is proportional to the distance moved by pinch gesture  from start point. |
| 10. | Volume Control | The rate of increase/decrease of volume is proportional  to the distance moved by pinch gesture from start point. |

|  |  |  |
| --- | --- | --- |
| 11. | Launch/Stop gesture recognition | Proton launch Gesture Recognition Proton stop Gesture Recognition |
| 12. | Google Search | Proton search {text\_you\_wish\_to\_search} |
| 13. | Find location | Proton Find a Location {location\_wanted to\_find} |
| 15. | Copy/Paste | Proton copy  Proton paste |
| 16. | Sleep/wake Proton | Proton bye Proton wakeup |
| 17. | File navigation | Proton list files / Proton list Proton open {file\_number} Proton go back / Proton back |

## SYSTEM FLOW DIAGRAM

System flow diagrams visually represent systems and processes that would be hard to describe through text. A system flow diagram represents the flow of a process in an easy and readable manner. The user shows the gestures in front of the camera. The nodes of the hand are detected using the mediapipe modules and it is compared with the gesture dictionary that is already processed. The gesture is then identified and the action is executed.

user

Hand detection

Voice Assistant

Pre processing

Command

Feature Extraction

Gesture Dictionary

Recognition

Execution

Fig 4.3 System Flow Diagram of Virtual Mouse and Voice Assistant

14

## USE CASE DIAGRAM

A UML use case diagram form of system requirements for a new software program which is under development. Use case diagram specifies the expected behavior of making the software happen. Use cases once specified can be denoted both textual and visual representation (i.e. use case diagram). The main motive of a use case diagram is that it helps us design a system from the end user’s perspective.

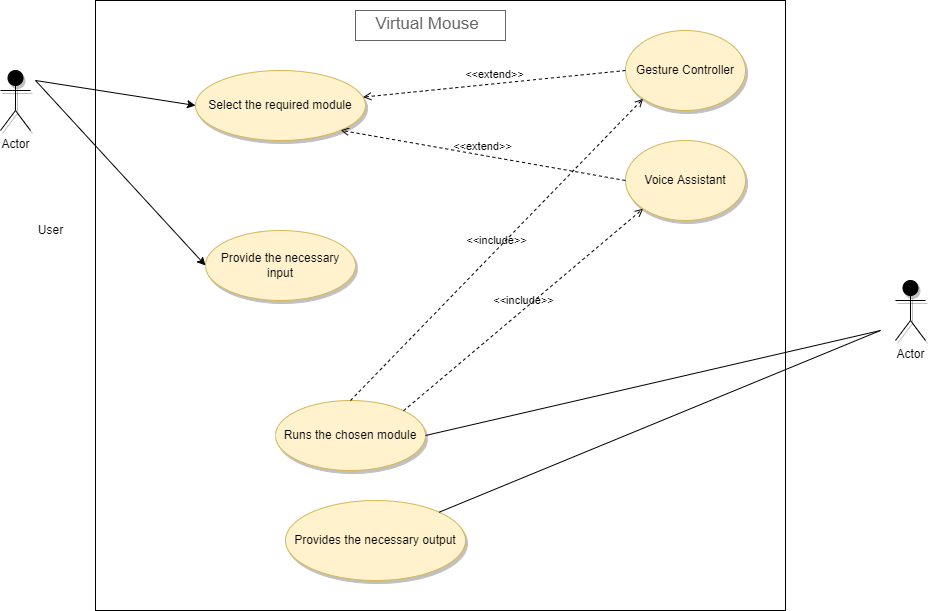


Fig 4.4 Use case Diagram of Virtual Mouse & Voice assistant

## CLASS DIAGRAM

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. It represents the static view of an application. It is also known as a structural diagram. In the above representation, there are five classes to demonstrate the virtual mouse. The corresponding methods are labelled in their respective classes.

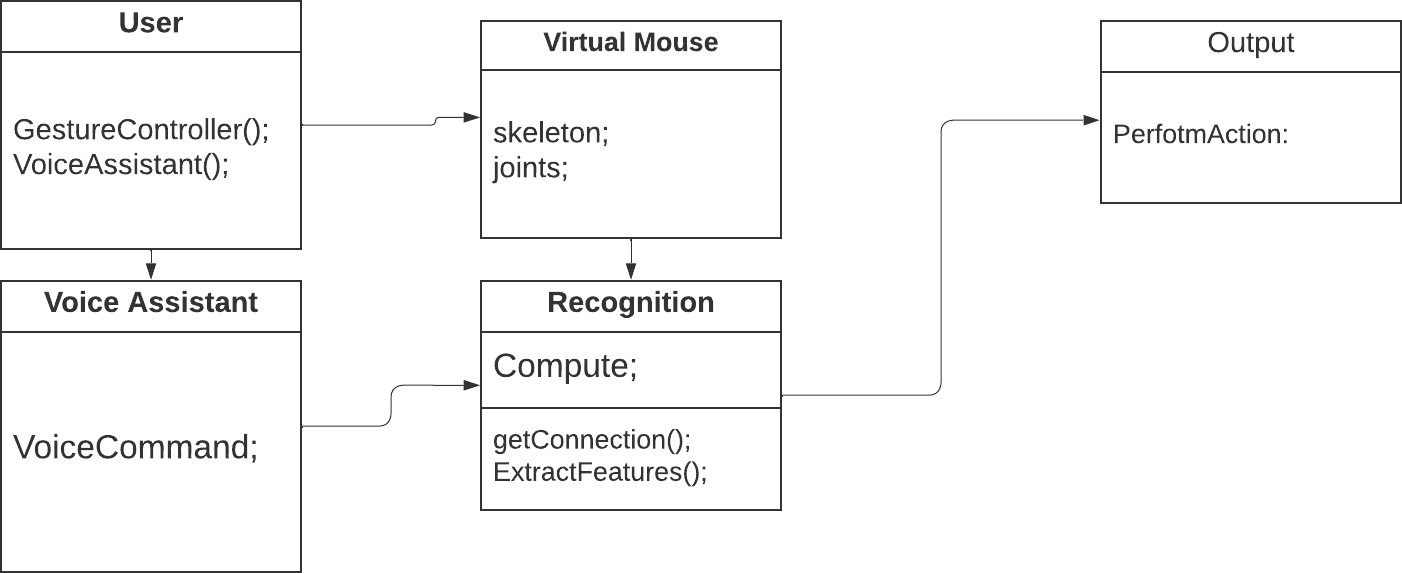


Fig 4.5 Class diagram of Virtual Mouse & Voice assistant

.

## ACTIVITY DIAGRAM

An activity diagram captures the dynamic behavior of the system. It portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed. The user gives the gestures as input through the webcam. The nodes of the hands are detected and compared with the predefined gestures and the necessary actions are executed. The input can also be given through the other module which is the voice assistant. In this module the input is given through voice commands.

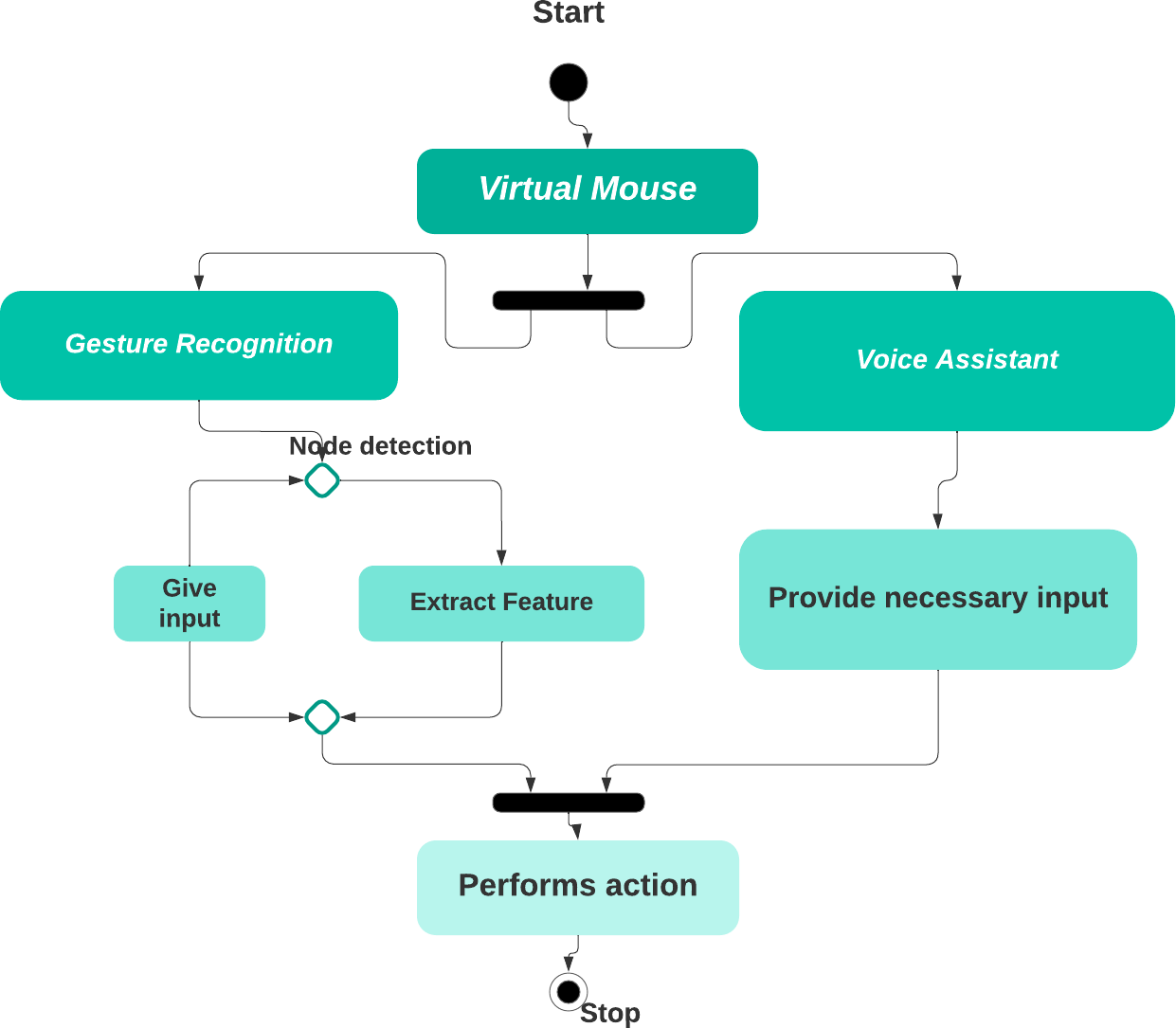


Fig 4.6 Activity diagram of Virtual Mouse & Voice assistant

## SEQUENCE DIAGRAM

Sequence diagrams explains the sequential flow of the process, in this diagram when the web cam recognizes user’s hand it extracts the feature of the node and executes according to the action provided by the user and for voice assistant it runs according to the voice input provided by the user only if it is a valid command.

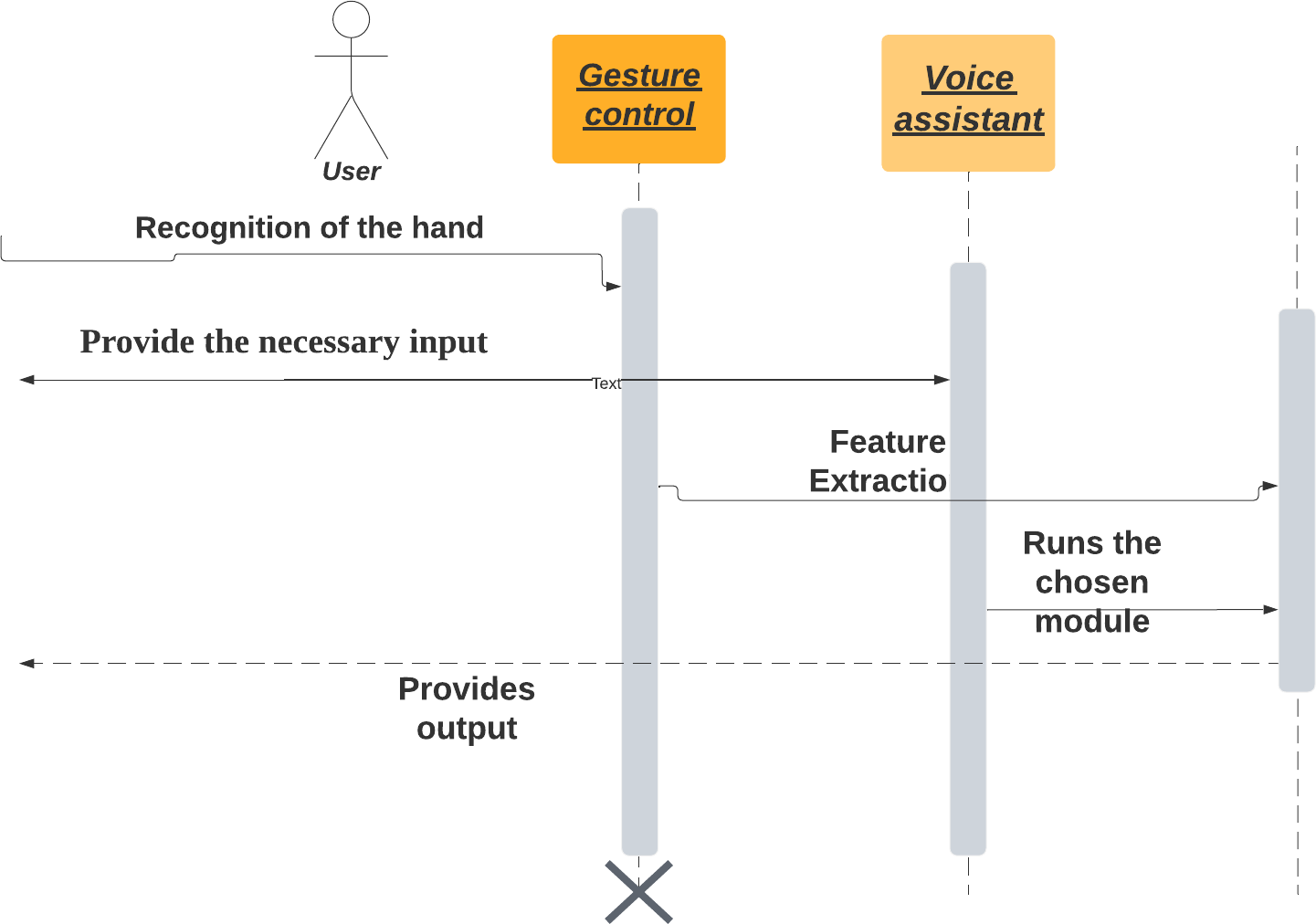


Fig 4.7 Sequence diagram of Virtual Mouse & Voice assistant

18

## CHAPTER 5 SYSTEM ARCHITECTURE

## SYSTEM ARCHITECTURE OVERVIEW

When the user activates the voice assistant or virtual mouse, it checks to see if the instructions are browser or system commands and then executes the instructions. If the user wishes to quit the voice assistant, standard exit instructions will be effective.

VOICE ASSISTANT

BROWSER COMMANDS

SYSTEM COMMANDS

Voice assistant

**if**

VALID COMMANDS

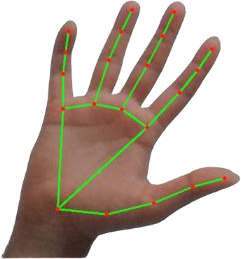
PERFORMS THE ACTIONS

Virtual mouse

**STOP**

INVALID COMMANDS

EXECUTION OF ACTION PERFORMED BY USER



FEATURE EXTRACTION

Receives image through webcam

Locate Nodes in the hand

Gesture Library

Fig 5.1 System architecture

19

## MODULE DESIGN SPECIFICATION

**Launching gesture recognition –** The user can also launch the gesture recognition system by proton. For that the user should give a command of “ Proton Launch Gesture Recognition”.

**Hand Detection Module –** In this module, the images of the hand are constantly captured by the OpenCV by initializing a holistic model. A webcam is used for capturing the images of the hand. For detecting the nodes of the hand precisely, the image of the hand should be within the frame. The nodes are detected simultaneously in accordance with the movements of the hand. So the user has to ensure that their hand is within the frame while using virtual mouse.

**Image Preprocessing Module –** The detected hand image will be processed into nodes to extract the feature requested by the user. Image data preprocessing, which converts image data into a form that allows machine learning algorithms to solve it. It is often used to increase a model’s accuracy, as well as reduce its complexity. While detecting it will detect each and every nodes of the hand to confirm the accuracy of the action to be performed.

**Feature Extraction Module –** After detecting the nodes of the hand, it will extract features from the gesture dictionary for the action to be performed. The gesture dictionary contains the libraries for the features like right click, left click, double click, drag folder, select multiple folders, brightness control, volume control, neutral gesture.

**Extraction of Features from Gesture Dictionary –** The exact feature for the detected node according to the user will be available in the pre-defined gesture dictionary. If the user wants to drag a folder, that particular gesture will be extracted with the help of the gesture libraries.

**Execution Module**– By recognizing those features, the virtual mouse will perform according to the provided action by the used and the voice assistant will execute according to the command given.

**Proton Activation Module –** The voice assistant Proton can be activated by the command python proton.py. Voice assistant Proton can do the tasks by waking the voice assistant by the voice command “Proton wake up”. After activating proton, the user can interact with it by giving voice command with proton as first word in every sentence. For example, “Proton what is today’s date”, “Proton find location”.

**Speech Recognition-** This module is used for the conversion of text to speech in a program. It works offline only. This module uses libraries such as sapi5 and pyttsx3 to recognize the commands given by the user. We will give an audio using microphone for speech recognizing. This module is used to make a query and give reply in the voice assistant.

## ALGORITHM

**Hand gesture recognition algorithm**

Hand gesture recognition algorithm that combines the hand-type adaptive algorithm and effective-area ratio based on feature matching.

The algorithm consists of

* + - Image acquisition,
    - Pre-processing,
    - Finger detection,
    - Gesture recognition.

These are overall steps involved in a successful hand gesture recognition algorithm.

## CNN Model

Convolutional Neural Network Model for classification among the desired mouse operations. The users are going to be allowed to regulate a number of the pc cursor functions with their hand gestures. Primarily, a user can perform left clicks, right clicks, and double clicks, scrolling up or down using their hand in several gestures. This technique captures frames employing a webcam or built-in cam and processes the frames to make them track-able and then recognizes different gestures made by users and perform the mouse functions. Therefore the proposed mouse system eliminates device dependency so as to use a mouse.

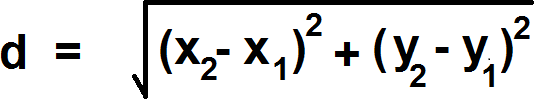
Gesture recognition is accomplished using a CNN model which is trained on hundreds of images to classify the gestures provided by the user among the known classes. The model is trained to classify the hand gestures into 5 classes. The classes are left click, left double click, right click, scroll down, scroll up. Each gesture is associated with a specific mouse operation. As soon as the trained CNN-model classifies the hand gesture of the user, the operation of the mouse associated with the gesture gets executed i.e if the hand gesture of the user is classified by the CNN model as left click, the left click

operation of the mouse gets executed.

**Recurrent neural networks (RNN) Algorithm**

RNN that remembers its input, due to an internal memory, which makes it perfectly suited for Voice assistant(proton) that involve sequential data.

The combination of CNN and RNN is a common and effective network structure for this task. Especially, we use 3DCNN in CNN part and ConvLSTM in RNN part. We divide the video into multiple temporal segments by average and compress each segment into one feature map by pooling layer.

The detected fingers are classified into their respective finger count. The gestures are classified using the maximum distance between the centroid of the two fingers determined in the finger detection process. The distance between each centroid of the two fingers is determined using the equidistance.

Where, ( X1 , Y1 ) = Coordinate of a pixel ( X2 , Y2 ) = Coordinate of a pixel

The maximum distance will be divided by two to obtain the radius of the circle. A circle will be drawn in the image, and it will be removed (palm) as we obtain the radius and center of the circle. Moreover, the radius should times by 1.26 to make the circle bigger so that only the fingers can be obtained. The number of an object inside the image will be calculated and displayed.

The algorithm was tested using different parameters to verify the performance of the hand gesture recognition algorithm. The parameters that chosen in this experiment were light intensity, size, noise, and effect of selective hand accessories.

## Capturing the Video and Processing

Virtual mouse system uses the webcam where each frame is captured till the termination of the program. Video frames are processed from BGR to RGB color space to find the hands in the video frame by frame as shown in the following code:

def findHands(self, img, draw = True):

imgRGB = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB) Self.results = self.hands.process(imgRGB)

## MediaPipe framework

For the purpose of detection of hand gestures and hand tracking, the MediaPipe framework is used, and OpenCV library is used for computer vision. The algorithm makes use of the machine learning concepts to track and recognize the hand gestures and hand tip. The involving steps in our proposed system uses MediaPipe framework as pipeline structure configuration. This pipeline structure create and run in various platforms which allowing scalability in mobile and desktop system also.

## Single-shot detector model:

Single shot detector model is used for detecting and recognizing a hand or palm in real time. The single-shot detector model is used by the MediaPipe. First, in the hand detection module, it is first trained for a palm detection model because it is easier to train palms. Furthermore, the non maximum suppression works significantly better on small objects such as palms or fists. A model of hand landmark consists of locating 21 joint or knuckle co-ordinates in the hand region.

## 6.1 CODING

**CHAPTER 6**

## SYSTEM IMPLEMENTATION

Gesture\_Controller\_Gloved.py import numpy as np

import cv2

import cv2.aruco as aruco import os

import glob import math import pyautogui import time

class Marker:

def init (self, dict\_type = aruco.DICT\_4X4\_50, thresh\_constant = 1): self.aruco\_dict = aruco.Dictionary\_get(dict\_type)

self.parameters = aruco.DetectorParameters\_create() self.parameters.adaptiveThreshConstant = thresh\_constant self.corners = None # corners of Marker

self.marker\_x2y = 1 # width:height ratio self.mtx, self.dist = Marker.calibrate()

def calibrate():

criteria = (cv2.TERM\_CRITERIA\_EPS + cv2.TERM\_CRITERIA\_MAX\_ITER, 30, 0.001)

objp = np.zeros((6\*7,3), np.float32) objp[:,:2] = np.mgrid[0:7,0:6].T.reshape(-1,2) objpoints = [] # 3d point in real world space

imgpoints = [] # 2d points in image plane.

Path = os.path.dirname(os.path.abspath( file )) p1 = path + r’\calib\_images\checkerboard\\*.jpg’ images = glob.glob(p1)

for fname in images:

img = cv2.imread(fname)

gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

ret, corners = cv2.findChessboardCorners(gray, (7,6),None) if ret == True:

objpoints.append(objp)

corners2 = cv2.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria) imgpoints.append(corners2)

img = cv2.drawChessboardCorners(img, (7,6), corners2,ret)

ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, gray.shape[::-1],None,None)

#mtx = [[534.34144579,0.0,339.15527836],[0.0,534.68425882,233.84359493],[0.0,0.0,1.0]]

#dist = [[-2.88320983e-01, 5.41079685e-02, 1.73501622e-03, -2.61333895e-04, 2.04110465e-01]]

return mtx, dist

def detect(self, frame):

gray\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY) self.corners, ids, rejectedImgPoints = aruco.detectMarkers(gray\_frame,

self.aruco\_dict, parameters = self.parameters) if np.all(ids != None):

rvec, tvec = aruco.estimatePoseSingleMarkers(self.corners, 0.05, self.mtx, self.dist)

else:

self.corners = None def is\_detected(self):

if self.corners: return True

return False

def draw\_marker(self, frame): aruco.drawDetectedMarkers(frame, self.corners)

def ecu\_dis(p1, p2):

dist = np.sqrt((p1[0]-p2[0])\*\*2 + (p1[1]-p2[1])\*\*2) return dist

def find\_HSV(samples): try:

color = np.uint8([ samples ]) except:

color = np.uint8([ [[105,105,50]] ])

hsv\_color = cv2.cvtColor(color,cv2.COLOR\_RGB2HSV) #print( hsv\_color )

return hsv\_color

def draw\_box(frame, points, color=(0,255,127)): if points:

frame = cv2.line(frame, points[0], points[1], color, thickness=2, lineType=8)

#top

frame = cv2.line(frame, points[1], points[2], color, thickness=2, lineType=8)

#right

frame = cv2.line(frame, points[2], points[3], color, thickness=2, lineType=8) #bottom

frame = cv2.line(frame, points[3], points[0], color, thickness=2, lineType=8)

#left

def in\_cam(val, type\_): if type\_ == ‘x’:

if val<0: return 0

if val>GestureController.cam\_width: return GestureController.cam\_width

elif type\_ == ‘y’: if val<0:

return 0

if val>GestureController.cam\_height: return GestureController.cam\_height

return val class ROI:

def init (self, roi\_alpha1=1.5, roi\_alpha2=1.5, roi\_beta=2.5, hsv\_alpha = 0.3, hsv\_beta = 0.5, hsv\_lift\_up = 0.3):

self.roi\_alpha1 = roi\_alpha1 self.roi\_alpha2 = roi\_alpha2 self.roi\_beta = roi\_beta self.roi\_corners = None self.hsv\_alpha = hsv\_alpha self.hsv\_beta = hsv\_beta self.hsv\_lift\_up = hsv\_lift\_up self.hsv\_corners = None

self.marker\_top = None

self.glove\_hsv = None

def findROI(self, frame, marker): rec\_coor = marker.corners[0][0]

c1 = (int(rec\_coor[0][0]),int(rec\_coor[0][1]))

c2 = (int(rec\_coor[1][0]),int(rec\_coor[1][1]))

c3 = (int(rec\_coor[2][0]),int(rec\_coor[2][1]))

c4 = (int(rec\_coor[3][0]),int(rec\_coor[3][1])) try:

marker.marker\_x2y = np.sqrt((c1[0]-c2[0])\*\*2 + (c1[1]-c2[1])\*\*2) / np.sqrt((c3[0]-c2[0])\*\*2 + (c3[1]-c2[1])\*\*2)

except:

marker.marker\_x2y = 999.0

#mid-point of top line of Marker cx = (c1[0] + c2[0])/2

cy = (c1[1] + c2[1])/2

self.marker\_top = [cx, cy]

l = np.absolute(ecu\_dis(c1,c4)) try:

slope\_12 = (c1[1]-c2[1])/(c1[0]-c2[0]) except:

slope\_12 = (c1[1]-c2[1])\*999.0 + 0.1

try:

slope\_14 = -1 / slope\_12 except:

slope\_14 = -999.0

if slope\_14 < 0: sign = 1

else:

sign = -1

bot\_rx = int(cx + self.roi\_alpha2 \* l \* np.sqrt(1/(1+slope\_12\*\*2)))

bot\_ry = int(cy + self.roi\_alpha2 \* slope\_12 \* l \* np.sqrt(1/(1+slope\_12\*\*2)))

bot\_lx = int(cx – self.roi\_alpha1 \* l \* np.sqrt(1/(1+slope\_12\*\*2)))

bot\_ly = int(cy – self.roi\_alpha1 \* slope\_12 \* l \* np.sqrt(1/(1+slope\_12\*\*2)))

top\_lx = int(bot\_lx + sign \* self.roi\_beta \* l \* np.sqrt(1/(1+slope\_14\*\*2))) top\_ly = int(bot\_ly + sign \* self.roi\_beta \* slope\_14 \* l \*

np.sqrt(1/(1+slope\_14\*\*2)))

top\_rx = int(bot\_rx + sign \* self.roi\_beta \* l \* np.sqrt(1/(1+slope\_14\*\*2))) top\_ry = int(bot\_ry + sign \* self.roi\_beta \* slope\_14 \* l \*

np.sqrt(1/(1+slope\_14\*\*2)))

bot\_lx = in\_cam(bot\_lx, ‘x’) bot\_ly = in\_cam(bot\_ly, ‘y’) bot\_rx = in\_cam(bot\_rx, ‘x’) bot\_ry = in\_cam(bot\_ry, ‘y’) top\_lx = in\_cam(top\_lx, ‘x’) top\_ly = in\_cam(top\_ly, ‘y’) top\_rx = in\_cam(top\_rx, ‘x’) top\_ry = in\_cam(top\_ry, ‘y’)

self.roi\_corners = [(bot\_lx,bot\_ly), (bot\_rx,bot\_ry), (top\_rx,top\_ry), (top\_lx,top\_ly)]

def find\_glove\_hsv(self, frame, marker): rec\_coor = marker.corners[0][0]

c1 = (int(rec\_coor[0][0]),int(rec\_coor[0][1]))

c2 = (int(rec\_coor[1][0]),int(rec\_coor[1][1]))

c3 = (int(rec\_coor[2][0]),int(rec\_coor[2][1]))

c4 = (int(rec\_coor[3][0]),int(rec\_coor[3][1])) l = np.absolute(ecu\_dis(c1,c4))

try:

slope\_12 = (c1[1]-c2[1])/(c1[0]-c2[0]) except:

slope\_12 = (c1[1]-c2[1])\*999.0 + 0.1

try:

slope\_14 = -1 / slope\_12 except:

slope\_14 = -999.0 if slope\_14 < 0:

sign = 1 else:

sign = -1

bot\_rx = int(self.marker\_top[0] + self.hsv\_alpha \* l \* np.sqrt(1/(1+slope\_12\*\*2)))

bot\_ry = int(self.marker\_top[1] – self.hsv\_lift\_up\*l + self.hsv\_alpha \* slope\_12

\* l \* np.sqrt(1/(1+slope\_12\*\*2)))

bot\_lx = int(self.marker\_top[0] – self.hsv\_alpha \* l \* np.sqrt(1/(1+slope\_12\*\*2)))

bot\_ly = int(self.marker\_top[1] – self.hsv\_lift\_up\*l – self.hsv\_alpha \* slope\_12 \* l \* np.sqrt(1/(1+slope\_12\*\*2))

top\_lx = int(bot\_lx + sign \* self.hsv\_beta \* l \* np.sqrt(1/(1+slope\_14\*\*2))) top\_ly = int(bot\_ly + sign \* self.hsv\_beta \* slope\_14 \* l \*

np.sqrt(1/(1+slope\_14\*\*2)))

top\_rx = int(bot\_rx + sign \* self.hsv\_beta \* l \* np.sqrt(1/(1+slope\_14\*\*2))) top\_ry = int(bot\_ry + sign \* self.hsv\_beta \* slope\_14 \* l \*

np.sqrt(1/(1+slope\_14\*\*2)))

region = frame[top\_ry:bot\_ry , top\_lx:bot\_rx] b, g, r = np.mean(region, axis=(0, 1))

self.hsv\_glove = find\_HSV([[r,g,b]])

self.hsv\_corners = [(bot\_lx,bot\_ly), (bot\_rx,bot\_ry), (top\_rx,top\_ry), (top\_lx,top\_ly)]

def cropROI(self, frame):

pts = np.array(self.roi\_corners ## (1) Crop the bounding rect rect = cv2.boundingRect(pts) x,y,w,h = rect

ikiped = frame[y:y+h, x:x+w].copy()

## (2) make mask

pts = pts – pts.min(axis=0)

mask = np.zeros( ikiped.shape[:2], np.uint8) cv2.drawContours(mask, [pts], -1, (255, 255, 255), -1, cv2.LINE\_AA) ## (3) do bit-op

dst = cv2.bitwise\_and( ikiped, ikiped, mask=mask) ## (4) add the white background

bg = np.ones\_like( ikiped, np.uint8)\*255 cv2.bitwise\_not(bg,bg, mask=mask) kernelOpen = np.ones((3,3),np.uint8) kernelClose = np.ones((5,5),np.uint8)

hsv = cv2.cvtColor(dst, cv2.COLOR\_BGR2HSV) lower\_range = np.array([self.hsv\_glove[0][0][0]//1-5,50,50])

upper\_range = np.array([self.hsv\_glove[0][0][0]//1+5,255,255]) mask = cv2.inRange(hsv, lower\_range, upper\_range)

#mask = cv2.dilate(mask,kernelOpen,iterations = 1)

Opening =cv2.morphologyEx(mask,cv2.MORPH\_OPEN,kernelOpen) Closing =cv2.morphologyEx(Opening,cv2.MORPH\_CLOSE,kernelClose)

FinalMask = Closing return FinalMask

class Glove:

def init (self): self.fingers = 0

self.arearatio = 0

self.gesture = 0

def find\_fingers(self, FinalMask): conts,h=cv2.findContours(FinalMask,cv2.RETR\_EXTERNAL,cv2.CHAIN\_APPRO X\_NONE)

hull = [cv2.convexHullI for c in conts] try:

cnt = max(conts, key = lambda x: cv2.contourArea(x)) #approx the contour a little

epsilon = 0.0005\*cv2.arcLength(cnt,True) ikiped= cv2.approxPolyDP(cnt,epsilon,True)

#make convex hull around hand hull = cv2.convexHull(cnt)

#define area of hull and area of hand areahull = cv2.contourArea(hull) areacnt = cv2.contourArea(cnt)

#find the percentage of area not covered by hand in convex hull self.arearatio=((areahull-areacnt)/areacnt)\*100

#find the defects in convex hull with respect to hand hull = cv2.convexHull( ikiped, returnPoints=False) defects = cv2.convexityDefects( ikiped, hull)

except:

print(“No Contours found in FinalMask”)

# l = no. of defects l=0

try:

#code for finding no. of defects due to fingers for I in range(defects.shape[0]):

s,e,f,d = defects[I,0]

start = tuple( ikiped[s][0]) end = tuple( ikiped[e][0]) far = tuple( ikiped[f][0])

# find length of all sides of triangle

a = math.sqrt((end[0] – start[0])\*\*2 + (end[1] – start[1])\*\*2)

b = math.sqrt((far[0] – start[0])\*\*2 + (far[1] – start[1])\*\*2)

c = math.sqrt((end[0] – far[0])\*\*2 + (end[1] – far[1])\*\*2) s = (a+b+c)/2

ar = math.sqrt(s\*(s-a)\*(s-b)\*(s-c))

#distance between point and convex hull d=(2\*ar)/a

# apply cosine rule here

angle = math.acos((b\*\*2 + c\*\*2 – a\*\*2)/(2\*b\*c)) \* 57

# ignore angles > 90 and ignore points very close to convex hull(they generally come due to noise)

if angle <= 90 and d>30: l += 1

#cv2.circle(frame, far, 3, [255,255,255], -1)

#draw lines around hand cv2.line(FinalMask,start, end, [255,255,255], 2)

l+=1

except:

l = 0

print(“No Defects found in mask”) self.fingers = l

def find\_gesture(self, frame):

font = cv2.FONT\_HERSHEY\_SIMPLEX

self.gesture = 0

if self.fingers==1:

#cv2.putText(frame, str(int(arearatio)), (10,50), font, 2, (0,0,255), 3, cv2.LINE\_AA)

if self.arearatio<15:

cv2.putText(frame,’0’,(0,50), font, 2, (0,0,255), 3, cv2.LINE\_AA)

self.gesture = 0 elif self.arearatio<25:

cv2.putText(frame,’2 fingers’,(0,50), font, 2, (0,0,255), 3, cv2.LINE\_AA)

self.gesture = 2 else:

cv2.putText(frame,’1 finger’,(0,50), font, 2, (0,0,255), 3, cv2.LINE\_AA)

self.gesture = 1

elif self.fingers==2:

cv2.putText(frame,’2’,(0,50), font, 2, (0,0,255), 3, cv2.LINE\_AA)

self.gesture = 3

elif self.fingers==3:

#cv2.putText(frame,’3’,(0,50), font, 2, (0,0,255), 3, cv2.LINE\_AA)

elif self.fingers==4:

#cv2.putText(frame,’4’,(0,50), font, 2, (0,0,255), 3, cv2.LINE\_AA)

elif self.fingers==5:

#cv2.putText(frame,’5’,(0,50), font, 2, (0,0,255), 3, cv2.LINE\_AA) else :

# cv2.putText(frame,’reposition’,(10,50), font, 2, (0,0,255), 3, cv2.LINE\_AA)

class Tracker:

def init (self): self.tracker\_started = False self.tracker = None self.start\_time = 0.0

self.now\_time = 0.0 self.tracker\_bbox = None

def corners\_to\_tracker(self, corners):

csrt\_minX = int( min( [corners[0][0][0][0], corners[0][0][1][0],

corners[0][0][2][0], corners[0][0][3][0]] ))

csrt\_maxX = int( max( [corners[0][0][0][0], corners[0][0][1][0],

corners[0][0][2][0], corners[0][0][3][0]] ))

csrt\_minY = int( min( [corners[0][0][0][1], corners[0][0][1][1],

corners[0][0][2][1], corners[0][0][3][1]] ))

csrt\_maxY = int( max( [corners[0][0][0][1], corners[0][0][1][1],

corners[0][0][2][1], corners[0][0][3][1]] ))

self.tracker\_bbox = [csrt\_minX, csrt\_minY, csrt\_maxX-csrt\_minX, csrt\_maxY- csrt\_minY]

def tracker\_to\_corner(self, final\_bbox):

if self.tracker\_bbox == None: return None

final\_bbox = [[[1,2],[3,4],[5,6],[7,8]]]

final\_bbox[0][0] = [self.tracker\_bbox[0],self.tracker\_bbox[1]] final\_bbox[0][1] = [self.tracker\_bbox[0]+

self.tracker\_bbox[2],self.tracker\_bbox[1]] final\_bbox[0][2] = [self.tracker\_bbox[0]+

self.tracker\_bbox[2],self.tracker\_bbox[1] + self.tracker\_bbox[3]] final\_bbox[0][3] = [self.tracker\_bbox[0],self.tracker\_bbox[1]

+self.tracker\_bbox[3]]

return [np.array(final\_bbox, dtype = ‘f’)]

def CSRT\_tracker(self, frame):

if self.tracker\_bbox == None and self.tracker\_started == False: return

if self.tracker\_started == False: if self.tracker == None:

self.tracker = cv2.TrackerCSRT\_create() if self.tracker\_bbox != None:

try:

self.start\_time = time.time()

ok = self.tracker.init(frame, self.tracker\_bbox) self.tracker\_started = True

except:

print(“tracker.init failed”)

try:

ok, self.tracker\_bbox = self.tracker.update(frame) except:

ok = None print(“tracker.update failed”)

self.now\_time = time.time()

if self.now\_time-self.start\_time >= 2.0 :

#cv2.putText(frame, “Please posture your hand correctly”, (10,50), cv2.FONT\_HERSHEY\_SIMPLEX, 1,(0,0,255),1)

cv2.putText(frame,’Posture your hand correctly’,(10,10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.75, (0,0,255), 1, cv2.LINE\_AA)

#print(“tracking timeout”) self.tracker\_started = False self.tracker\_bbox = None return

if ok:

# Tracking success

p1 = (int(self.tracker\_bbox[0]), int(self.tracker\_bbox[1]))

p2 = (int(self.tracker\_bbox[0] + self.tracker\_bbox[2]), int(self.tracker\_bbox[1]

+ self.tracker\_bbox[3]))

cv2.rectangle(frame, p1, p2, (80, 255, 255), 2, 1) else :

# Tracking failure self.tracker\_started = False

cv2.putText(frame, “Tracking failure detected”, (100,80), cv2.FONT\_HERSHEY\_SIMPLEX, 0.75,(0,0,255),2)

print(“Tracking failure detected”) #reintiallize code to tackle tracking failure

class Mouse:

def init (self): self.tx\_old = 0

self.ty\_old = 0 self.trial = True self.flag = 0

def move\_mouse(self,frame,position,gesture): (sx,sy)=pyautogui.size()

(camx,camy) = (frame.shape[:2][0],frame.shape[:2][1]) (mx\_old,my\_old) = pyautogui.position()

Damping = 2 # Hyperparameter we will have to adjust tx = position[0]

ty = position[1] if self.trial:

self.trial, self.tx\_old, self.ty\_old = False, tx, ty

delta\_tx = tx – self.tx\_old delta\_ty = ty – self.ty\_old self.tx\_old,self.ty\_old = tx,ty

if (gesture == 3): self.flag = 0

mx = mx\_old + (delta\_tx\*sx) // (camx\*Damping) my = my\_old + (delta\_ty\*sy) // (camy\*Damping) pyautogui.moveTo(mx,my, duration = 0.1)

elif(gesture == 0): if self.flag == 0:

pyautogui.doubleClick() self.flag = 1

elif(gesture == 1): print(‘1 Finger Open’)

class GestureController:

gc\_mode = 0 pyautogui.FAILSAFE = False f\_start\_time = 0

f\_now\_time = 0

cam\_width = 0

cam\_height = 0 aru\_marker = Marker()

hand\_roi = ROI(2.5, 2.5, 6, 0.45, 0.6, 0.4)

glove = Glove() csrt\_track = Tracker() mouse = Mouse()

def init (self):

GestureController.cap = cv2.VideoCapture(0) if GestureController.cap.isOpened():

GestureController.cam\_width = int( GestureController.cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH) )

GestureController.cam\_height = int( GestureController.cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT) )

else:

print(“CANNOT OPEN CAMERA”)

GestureController.gc\_mode = 1 GestureController.f\_start\_time = time.time() GestureController.f\_now\_time = time.time()

def start(self): while (True):

## Proton.py

from datetime import date import time

import webbrowser import datetime

from pynput.keyboard import Key, Controller import pyautogui

import sys import os

from os import listdir

from os.path import isfile, join import smtplib

import ikipedia

import Gesture\_Controller

#import Gesture\_Controller\_Gloved as Gesture\_Controller import app

from threading import Thread

# Object Initialization today = date.today()

r = sr.Recognizer() keyboard = Controller()

engine = pyttsx3.init(‘sapi5’) engine = pyttsx3.init()

voices = engine.getProperty(‘voices’) engine.setProperty(‘voice’, voices[0].id)

# Variables file\_exp\_status = False

files =[] path = ‘’

is\_awake = True #Bot status

# Functions def reply(audio):

app.ChatBot.addAppMsg(audio)

print(audio) engine.say(audio) engine.runAndWait()

def wish():

hour = int(datetime.datetime.now().hour) if hour>=0 and hour<12:

reply(“Good Morning!”) elif hour>=12 and hour<18:

reply(“Good Afternoon!”) else:

reply(“Good Evening!”)

reply(“I am Proton, how may I help you?)

## CHAPTER 7 PERFORMANCE ANALYSIS

Performance analysis was done for ten times to perform a variety of quick gestures for the purpose of determining the accuracy of the detection. This data was recorded and was used for the analysis so that we could demonstrate that our model is more compatible with actual applications and more user friendly.

We are going to make the assumption that X is the number of fingertips that are displayed on the right hand. Each individual person acts out the gestures under the standard lighting conditions. The trails were performed with each gesture from 1 to 5 ten times, which resulted in a total of 600 gestures that were manually labelled as ground truth. The gestures ranged from mouse movement (X = 1) to left-click (X = 2) to right- click (X = 3 || X = 4) to no action (X = 5 || X = 0). Because we were concentrating on right-hand movement for ease of detection and precision, every participant was right handed.

The results of our experimental testing of our Virtual mouse system are presented in Table 1. The overall accuracy comes out to 96.13 percent on average. This performance is exceptionally high for an interface that is based on gestures made with the fingertips. As was to be expected, the highest accuracy was achieved with the less difficult gesture known as “mouse movement,” while the lowest accuracy was achieved with the more difficult gesture known as “right-click.” The accuracy of the ‘right-click’ gesture was decreased because fast fingertip tracking sometimes caused it to be confused with other gestures. The experiment also demonstrated that the findings did not significantly shift across a number of different resolutions.

Table 7.1 Accuracy Calculation of Virtual Mouse

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Gesture** | **Node**  **detection** | **Success** | **Failure** | **Accuracy(%)** |
| Mouse movement | 1 | 100 | 0 | 100 |
| Left Click | 2 | 97 | 3 | 97 |
| Right Click | 3 | 90 | 10 | 89 |
| Click and Drag | 4 | 88 | 12 | 85 |
| Double Click | 5 | 99 | 1 | 98.5 |
| Neutral  Gesture | 0 | 100 | 0 | 100 |
|  |  | **574** | **26** | **95** |

The result shows that there is no significant difference between the normal light and faint-light conditions during the tracking. This means that the system can work well with different light levels. The proposed method also performs well with changing backgrounds and tracking at longer distances.

In order to evaluate system performance, the accuracy of the system was measured using the equation.

Where,

## Accuracy = DF / TF × 100%

DF- number of successfully recognized operations TF -number of total operations.

The accuracy of our system using the above equation is 95%. Since the system uses webcam captured videos, the performance of the system may depend on illumination. Additionally, if there are other colored objects are present in the background, the system may produce an incorrect response. Although this issue can be minimized by configuring the threshold values and other device parameters, it is still advisable that the operating background should be light and there should not be any bright colored artifacts present in the background.

Additionally, on some low computing computers, the device could run slower because it performs a large number of complex calculations in a very short time. However, for an optimal system performance, a regular computer or laptop has the computational power needed. Another aspect is that the device will run slow if the camera’s resolution is too high. This problem can be solved by reducing image resolution.

An AI system utilizing computer vision to improve human-computer interaction is proposed in the proposed visual mouse AI system. Because of the limited number of data sets available, it is difficult to compare contradictory outcomes of testing of visual AI mouse systems. The experimental results indicated that this approach is a promising technique for Gesture recognition interfaces in real time.

## CHAPTER 8

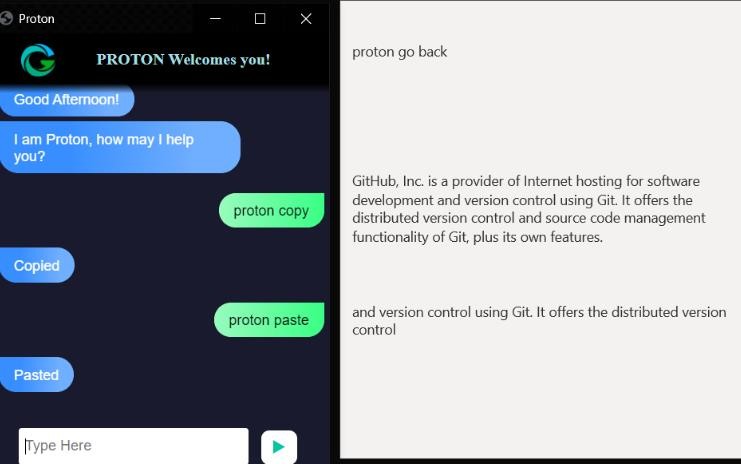
**CONCLUSION AND FUTURE ENHANCEMENT**

We have designed a system that will enable the physical mouse to be replaced with our own virtual mouse in the near future. The best way for a human to interact with a computer is through the use of a virtual mouse. The recognition of gestures is essential for the development of alternative modalities of interaction between humans and computers. It enables a more natural interaction between humans and machines than was previously possible. Our virtual mouse will be helpful and effective in minimizing the amount of clutter in the working environment. We think that our virtual mouse is the best place to begin the process of bringing artificial intelligence-based modernization to the whole world.

This virtual mouse is useful in many different areas, including augmented reality, computer graphics, computer gaming, prosthetics, and biomedical instrumentation, among others. Our system has been expanded with the addition of Digital Canvas, which is gaining popularity among artists, by which the artist could create 2D or 3D images using the Virtual Mouse technology using the hand as brush and a Virtual Reality kit or a monitor as display set which is becoming increasingly well-liked among AI devotees. In the future, we intend to incorporate the software of our virtual mouse into an application that will be used by regular people all over the world. We will be focusing on developing more efficient data fusion methods utilizing hand pose features and more effective networks for dynamic hand gesture recognition. In addition, we will add on-line learning or incremental learning to our method for unknown hand gestures. We also intend to expand our system to handle more gestures and interact with other smart environments

## APPENDICES

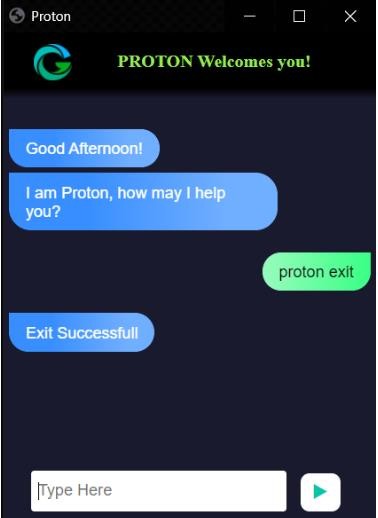
## A1. SAMPLE SCREENS



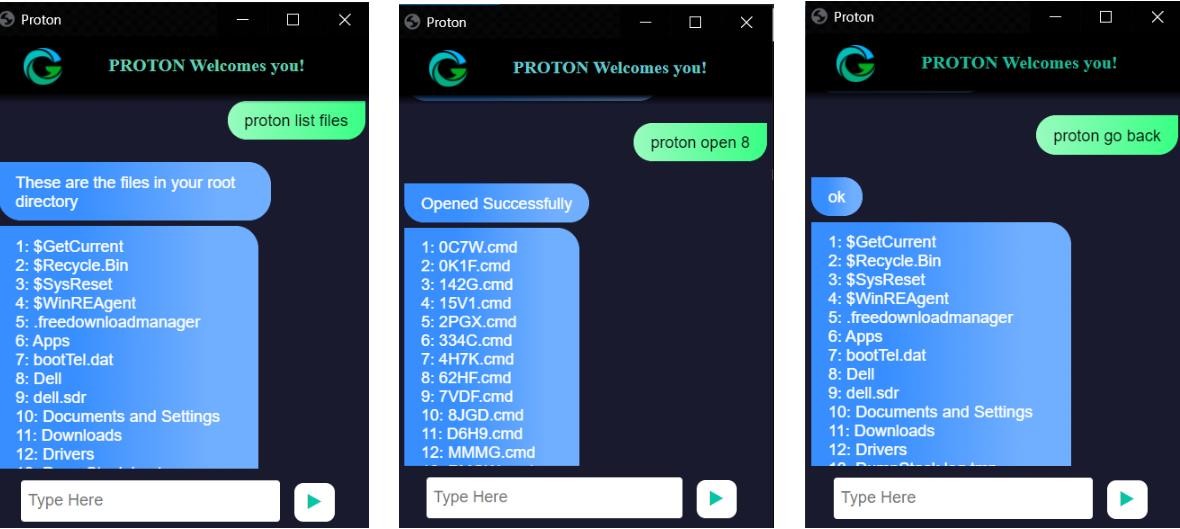
## Fig A1.1 Copy and Paste in voice assistant



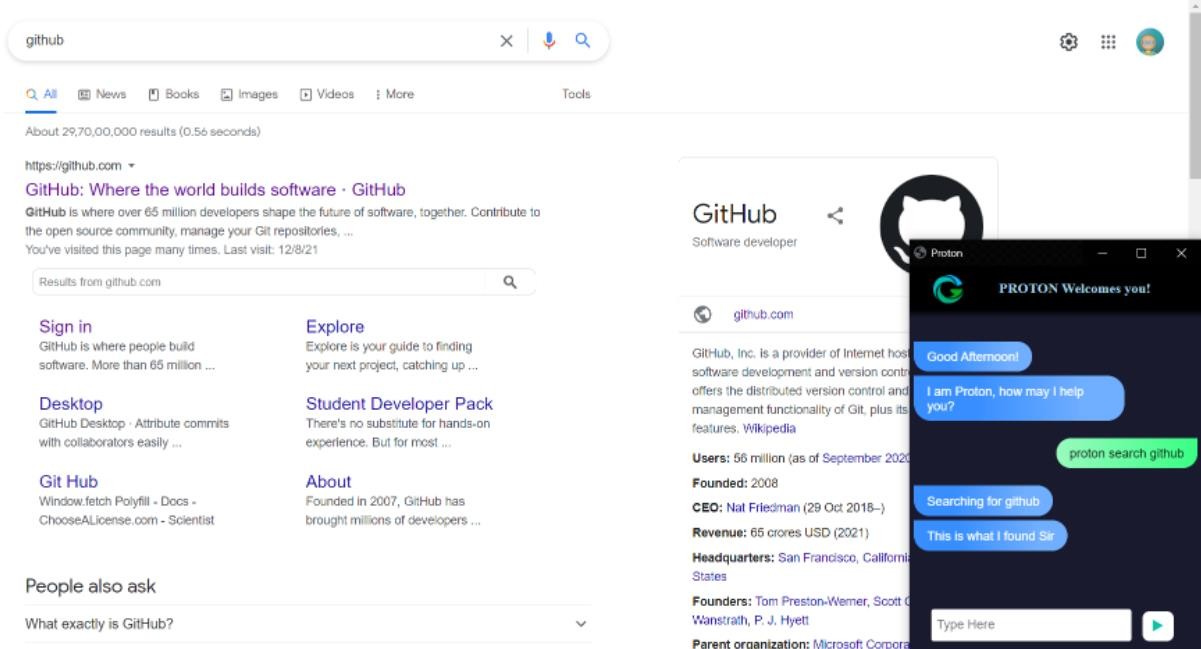
**Fig A1.2 Date and Time in voice assistant**



## Fig A1.3 Exit of voice assistant



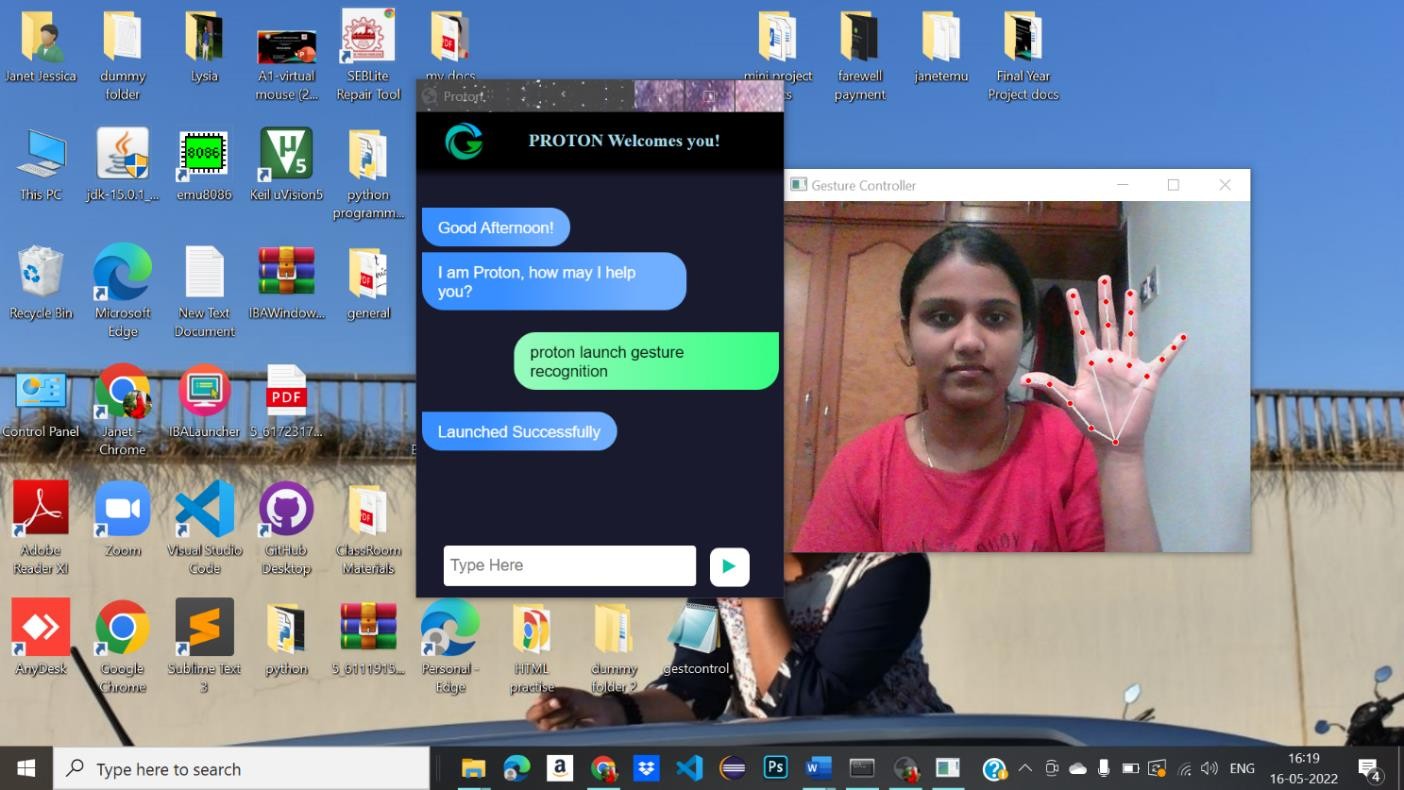
**Fig A1.4 File navigation in voice assistant**



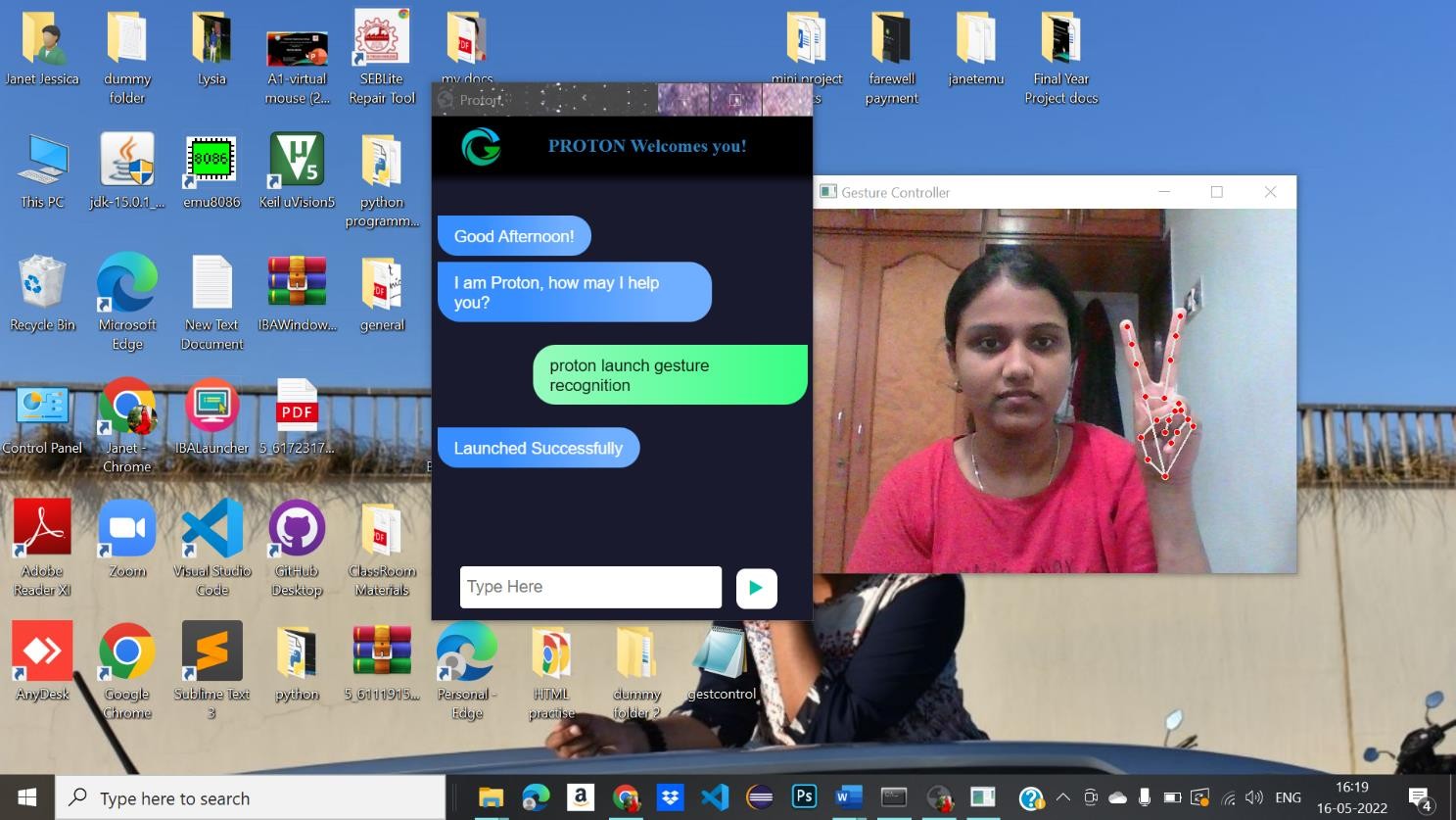
## Fig A1.5 Google Search in voice assistant



**Fig A1.6 Launch/Stop Gesture Recognition**



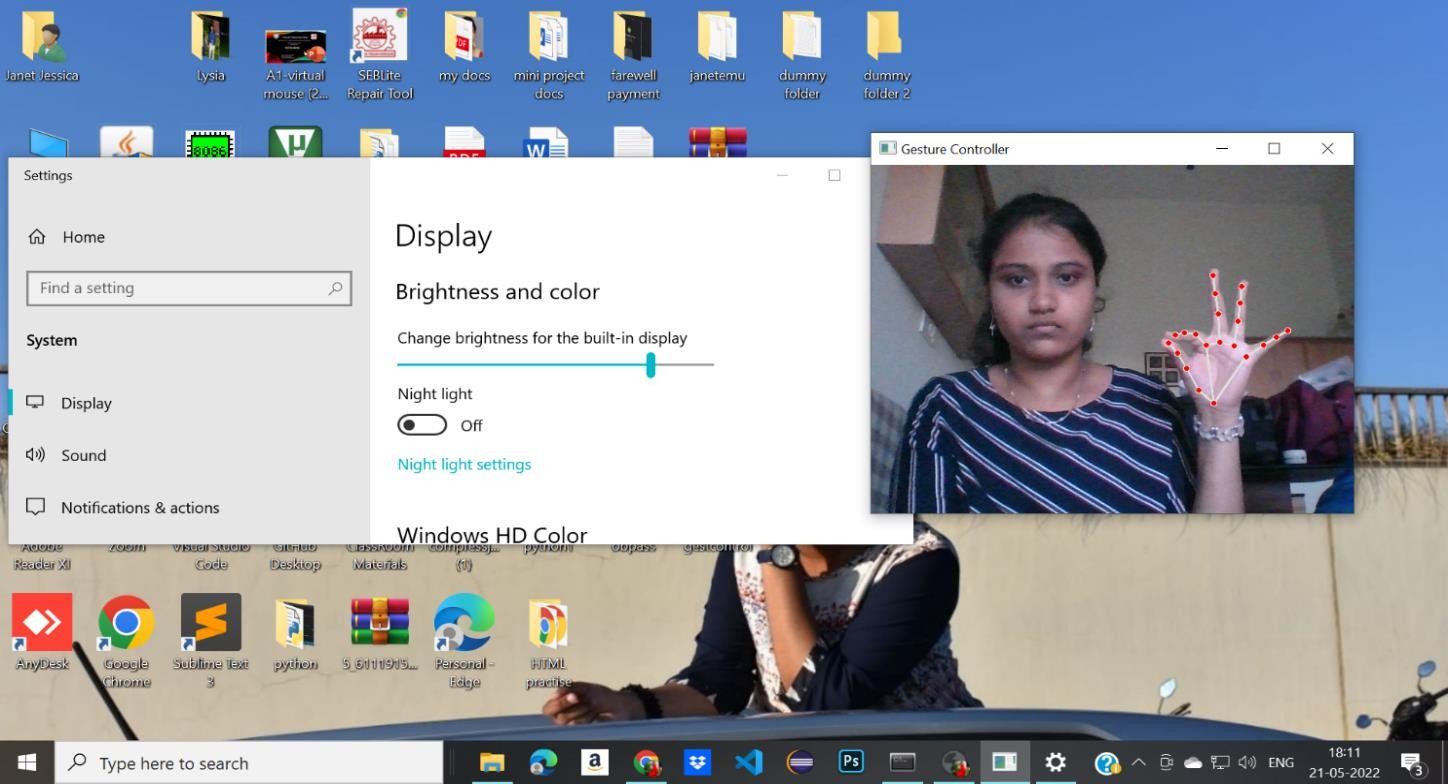
## Fig A1.7 Neutral Gesture



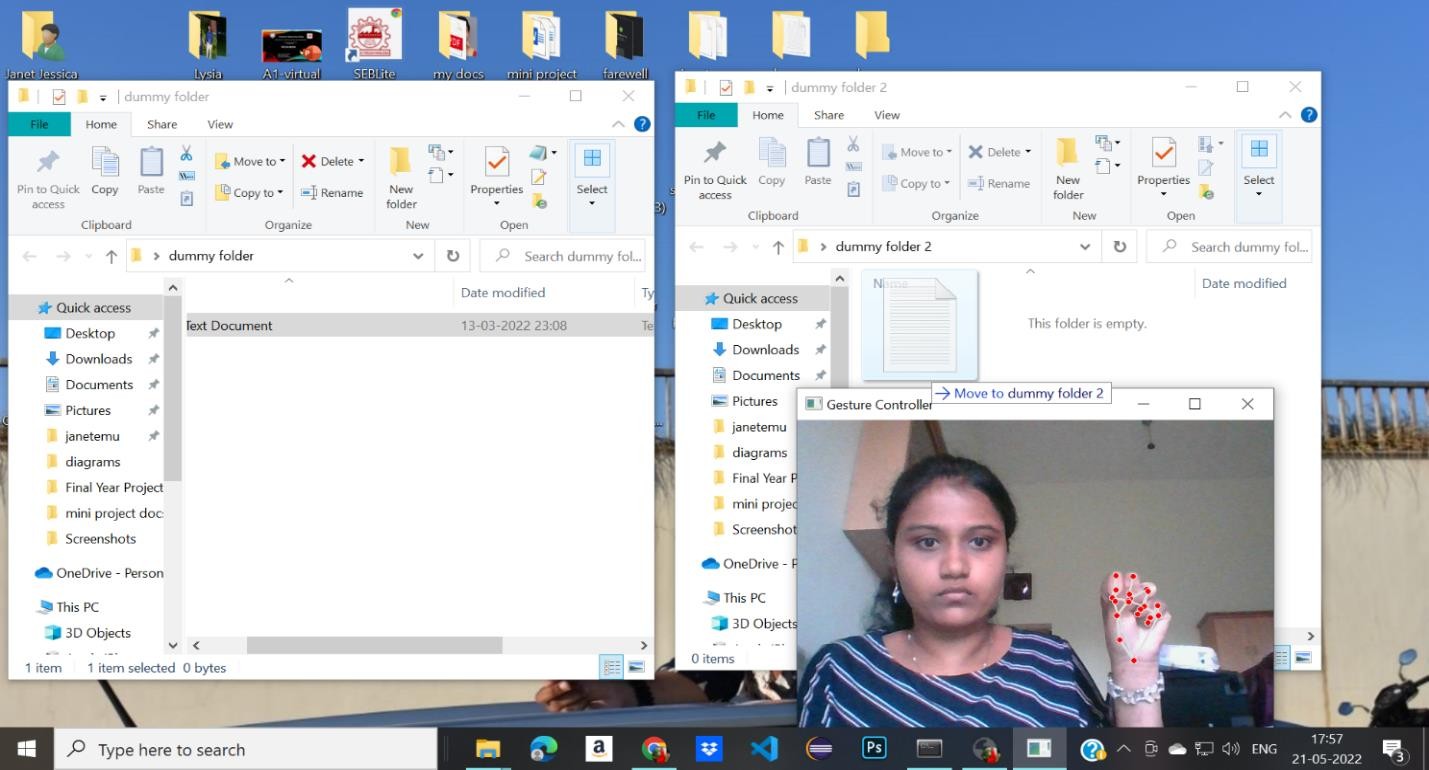
**Fig A1.8 Move cursor**



## Fig A1.9 Right click



**Fig A1.10 Brightness control**



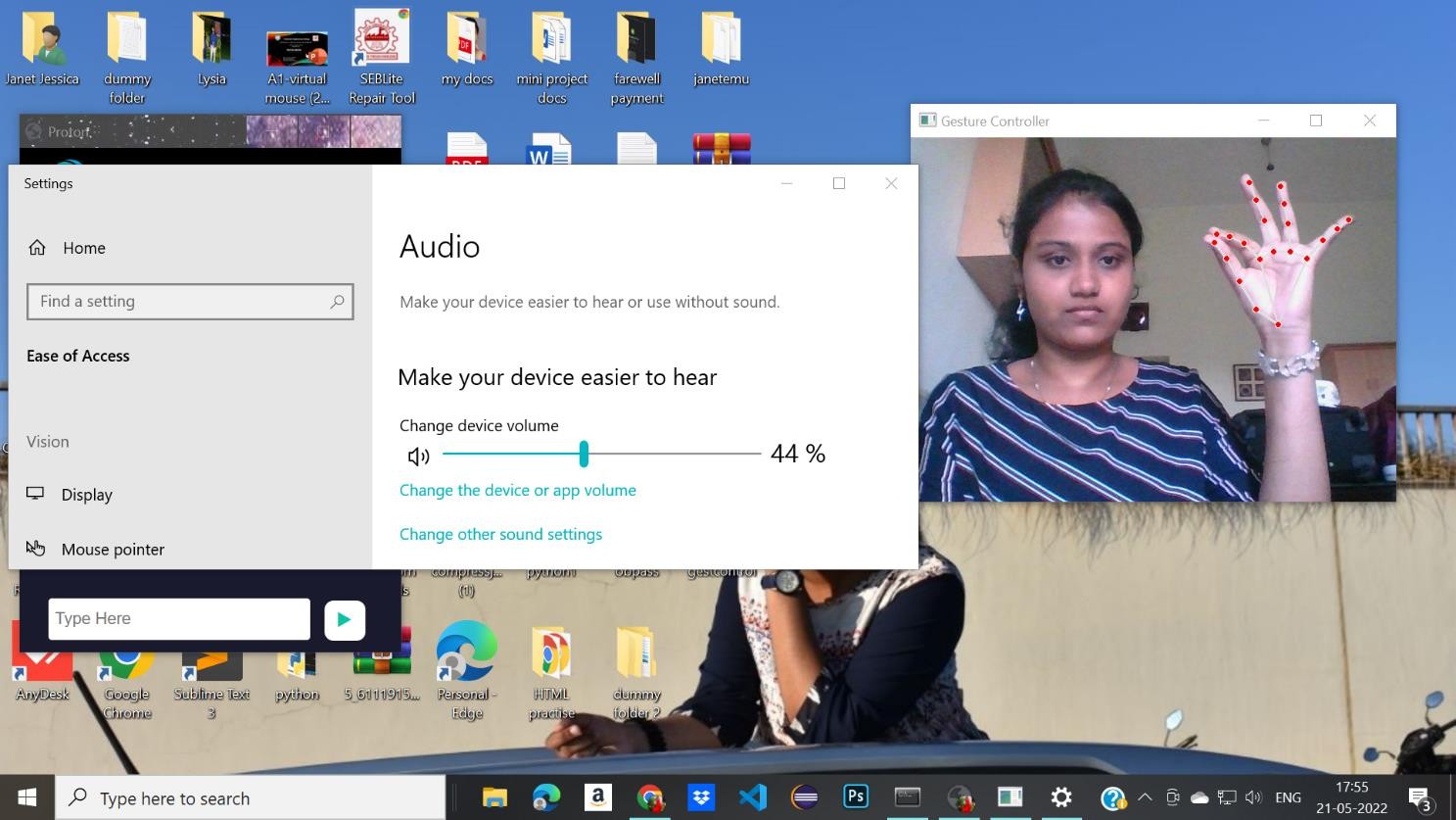
## Fig A1.11 Click and Drag



**Fig A1.12 Double click**



## Fig A1.13 Left click



**Fig A1.14 Volume control**

## REFERENCES

1. Banda Aneela , Narlanka Vasuki, Rebelli Sai Sahana, Charupalli Sunil Kumar, (2021), ”Implementing a Real Time Virtual Mouse System and Fingertip Detection based on Artificial Intelligence”. (IEEE), pp. 6299–6308.
2. [Sherin](https://ieeexplore.ieee.org/author/37086874147) [Mohammed](https://ieeexplore.ieee.org/author/37086874147) [Sali](https://ieeexplore.ieee.org/author/37086874147) [Shajideen](https://ieeexplore.ieee.org/author/37086874147); [VH](https://ieeexplore.ieee.org/author/37086054925) [Preetha](https://ieeexplore.ieee.org/author/37086054925)(2019) “Hand Gestures - Virtual Mouse for Human Computer Interaction” pp. 5239–2301.
3. Vantukala Vishnu Teja Reddy, Thumma Dhyanchand , Galla Vamsi Krishna , Satish Maheshwaram (2019), “Virtual Mouse Control Using hand gloves ” (IEEE), pp. 1239–5239.
4. Tiwari, Siddhi Parkar, Shruti Gharat, Kinjal Patel, Prof. Khalil Pinjari, (2019) “Design and Development of Hand Gesture Based Virtual Mouse”, (IEEE) , pp. 8745–3219.
5. Kollipara Sai Varun, I. Puneeth, Dr. T. Prem Jacob, (2019) “Virtual Mouse Implementation using Open CV”, (IEEE), pp. 2341–9834.
6. Swati Tiwari, Siddhi Parkar, Shruti Gharat, Kinjal Patel, Prof. Khalil Pinjari, (2018), “Virtual mouse using sixth sense technology”, pp. 4312–7865.
7. Sherin Mohammed Sali Shajideen Preetha V H, YEAR: March 2018, “Virtual Mouse for Human Computer Interaction”, ISBN : pp. 7891–2308.
8. Changhyun jeon, oh-jin kwon, dongil shin, and dongkyoo shin, (2018), “Hand- Mouse Interface Using Virtual Monitor Concept for Natural Interaction (IEEE), pp. 6534–2354.
9. Itsuo Kumazawa, Toshihiro Kai, Yoshikazu Onuki, and Shunsuke Ono,(2017) “Measurement of 3D-Velocity by High-Frame-Rate Optical Mouse Sensors to Extrapolate 3D Position Captured”, (IEEE), pp. 7634-1254.
10. S. S. Rautaray, A. Agrawal, “Vision based hand gesture recognition for human computer interaction: a survey,” Artificial intelligence review., vol. 43, no. 1, pp. 1–54, 2017.
11. O. K. Oyedotun, A. Khashman, “Deep learning in vision-based static hand gesture recognition” Neural Computing and Applications., vol. 28, no. 12, pp. 3941–3951, 2017.
12. L. Zhang, G. Zhu, P. Shen, J. Song, “Learning spatiotemporal features using 3dcnn and convolutional lstm for gesture recognition,” Proceedings of the IEEE International Conference on Computer Vision (ICCV)., pp. 3120–3128,2017.

[13]Q. Miao, Y. Li, W. Ouyang, Z. Ma, X. Xu, “Multimodal gesture recognition based on the rescd network,” Proceedings of the IEEE International Conference on Computer Vision (ICCV)., pp. 3047– 3055,2017.

1. Y. Zhu, Z. Lan, S. Newsam, A. Hauptmann, “Hidden two-stream convolutional networks for action recognition,” Asian Conference on Computer Vision., pp. 363–378, 2018.
2. B. Fang, F. Sun, H. Liu, C. Liu, “3D human gesture capturing and recognition by the IMMU-based data glove” Neurocomputing., vol. 277, pp. 198–207, 2018.
3. O. Kopuklu, N. Kose, G. Rigoll, “Motion fused frames: Data level fusion strategy for hand gesture recognition,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR)., pp. 2103–211, 2018.
4. J. Hou, G. Wang, X. Chen, J. H. Xue, “Spatial-temporal attention resTCN for skeleton-based dynamic hand gesture recognition,” Proceedings of the European Conference on Computer Vision (ECCV)., pp. 1– 9, 2018.
5. J. C. Nunez, R. Cabido, J. J. Pantrigo, A. S. Montemayor, “Convolutional neural networks and long short-term memory for skeleton-based human activity and hand gesture recognition,” Pattern Recognition., vol. 76, pp. 80–94, 2018.
6. Upasana, V Monisha Joseph, Kanchana V.p , YEAR : May,2017 “Virtual Mouse with RGB Colored Tapes”, (IEEE), pp. 2319–5412.
7. A. Mhetar, B. K. Sriroop, A. G. S. Kavya, R. Nayak, R. Javali and K. V. Suma, "Virtual mouse," International Conference on Circuits, Communication, Control and Computing, Bangalore, pp. 69-72, 2018.
8. Q. Gao, J. Liu, Z. Ju, “Robust real-time hand detection and localization for space human C robot interaction based on deep learning” Neurocomputing., 2019.
9. Z. Cao, T. Simon, S. E. Wei, “Spatial-temporal attention res-TCN for skeleton- based dynamic hand gesture recognition,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)., pp. 7291–7299, 2019.
10. S. Woo, J. Park, J. Y. Lee, et al, “Cbam: Convolutional block attention module,” Proceedings of the European conference on computer vision (ECCV)., pp. 3–19, 2019.
11. J. Carreira, A. Zisserman, “Quo vadis, action recognition? a new model and the kinetics dataset,” proceedings of the IEEE Conference on Computer Vision and Pattern Recognition., 2019, pp. 6299–6308.
12. M. Abavisani, H. R. V. Joze, V. M.Patel, “Improving the performance of unimodal dynamic hand-gesture recognition with multimodal training,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition., pp. 1165–1174, 2020 .
13. Q. Huang, D. Yang, L. Jiang, et al, “A novel unsupervised adaptive learning method for long-term electromyography (EMG) pattern recognition,” Sensors., vol. 17, no. 6, pp. 1370, 2020.
14. M. Simao, N. Mendes, O. Gibaru, et al, “A review on electromyography decoding and pattern recognition for human-machine interaction,” IEEE Access., vol. 7, pp. 39564-39582, 2020.
15. P. Kaczmarek, T. Makowski, J. Tomczyski, “putEMGłA Surface Electromyography Hand Gesture Recognition Dataset,” Sensors., vol. 19, no. 16, pp. 3548, 2020
16. Q. Gao, J. Liu, Z. Ju, “Hand gesture recognition using multimodal data fusion and multiscale parallel convolutional neural network for humanCrobot interaction” Expert Systems., 2020.
17. Erdem, E. Yardimci, Y. Atalay, V. Cetin, A. E.“Computer vision-based mouse”, Acoustics, Speech, And Signal Processing, Proceedings. (ICASS). IEEE International Conference.
18. Hojoon Park, “A Method for Controlling the Mouse Movement using a Real-Time Camera”, Brown University, Providence, RI, USA, Department of computer science.
19. T. D. Grove, K. D. Baker and T. N. Tan, "color based object tracking," Proceedings. Fourteenth International Conference on Pattern Recognition (Cat. No.98EX170), Brisbane, Queensland, Australia, pp. 1442-1444 vol.2, 2020.
20. N. Vo, Q. Tran, T. B. Dinh, T. B. Dinh and Q. M. Nguyen, "An Efficient Human- Computer Interaction Framework Using Skin Color Tracking and Gesture Recognition," 2010 IEEE RIVF International Conference on Computing & Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), Hanoi, 2020, pp. 1-6.
21. D. A. Barhate and K. P. Rane, "A Survey of Fingertip Character Identification in Open-Air Using Image Processing and HCI," 2018 3rd International Conference for Convergence in Technology (I2CT), Pune, pp. 1-4, 2018.
22. J.Suh, M. Amjadi, I. Park and H. Yoo, "Finger motion detection glove toward human-machine interface," 2021 IEEE SENSORS, Busan, pp. 1-4,2021